PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/445 — A1

(11) International Publication Number: WO 00/43873

(43) International Publication Date: 27 July 2000 (27.07.00)

(21) International Application Number: PCT/US00/00490

(22) International Filing Date: 20 January 2000 (20.01.00)

(30) Priority Data:
60/116,682 20 January 1999 (20.01.99) US

(71) Applicant: MAKARIOS, Selene, K. [US/US]; 900 High School Way #2202, Mountain View, CA 94041 (US).

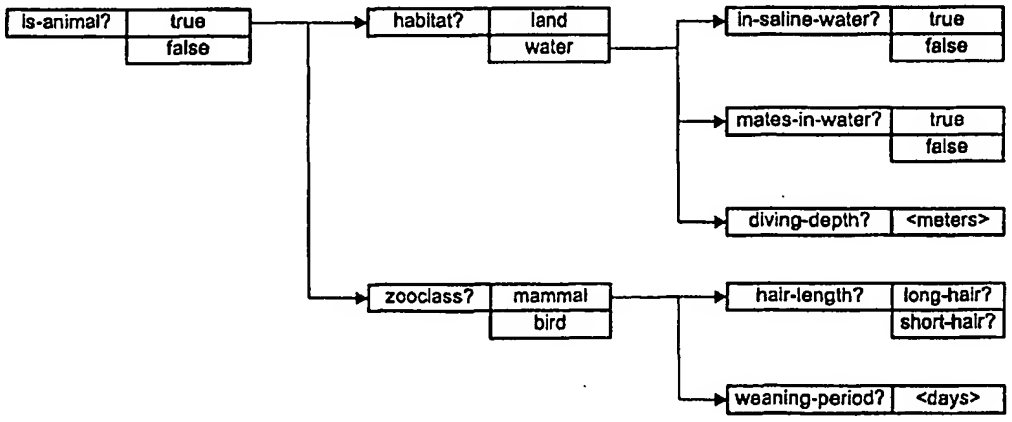
(72) Inventors: DOWNS, Heather, A.; 900 High School Way, #2202, Mountain View, CA 94041 (US). FITZWILSON, Robert, C.; 130 Tuscaloosa Avenue, Atherton, CA 94205 (US). OLSON, Perry, V.; 98 Stern Lane, Atherton, CA 94027 (US).

(74) Agents: BENGTSSON, W., Patrick et al.; Pillsbury Madison & Sutro, LLP, 1100 New York Avenue, N.W., Washington, DC 20005 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: ATTRIBUTE–ORIENTED PROGRAMMING–LANGUAGE CONSTRUCTS



(57) Abstract

Attribute–oriented programming–language constructs, a means for computer–programmatic implementation of software design based on attribute–maps where an attribute–map including attributes and applicability–predicates for defining a range of entities is shown in the figure. The attribute–maps are a modeling and diagramming technique for use in software design, which depict taxonomies of attributes relevant to a given software problem domain, and are particularly suited to domains which are knowledge–rich and complex, such as occur in finance or medicine.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Attribute-Oriented Programming-Language Constructs

FIELD OF THE INVENTION

The present invention relates to constructs for use within computer-programming languages. Historical examples of such constructs are those introduced with the disciplines of Structured Programming and Object-Oriented Programming. The present invention particularly relates to computer-programmatic implementation of software designs based on attribute-maps, which are a technique for creating models and representative diagrams of knowledge-rich domains, by systematic taxonornization of the relevant attributes of domain objects.

BACKGROUND OF THE INVENTION

Each new computer-programming paradigm has brought with it new programming-language constructs. Structured programming incorporated the use of the "for ... next" construct, the "if ... then ... else" construct, and the "while... do..." construct. Current-practice object-oriented programming

incorporated the programming-language constructs for class declaration and for the application of member functions.

The present invention improves upon the constructs of current-practice object-oriented programming by providing means for the computer-programmatic implementation of software designs created using attribute-maps. Attribute-maps are covered in a separate patent application, and the prior art pertaining to their invention is described therein. An example attribute-map is given in Figure 1, for use in later exposition.

SUMMARY OF THE INVENTION

The inventions described herein include:

1) An attribute-oriented programming-language construct termed the applies-if guard.

2) A technique of attribute-oriented method declaration using the applies-if guard.

3) An attribute-oriented programming-language construct termed the applies-if block.

In brief, the present invention (in at least one of its embodiments) includes a method for writing computer programs using a high-level computer language. The method includes the steps of: 1) defining an attribute; and 2) defining an applicability-predicate associated with the attribute, the applicability-predicate defining a range of entities to which the attribute applies.

Advantages of the invention will be set forth, in part, in the description that follows and, in part, will be understood by those skilled in the art from the

description herein. The advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims and equivalents.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments of the invention and, together with the description, serve to explain the principles of the invention.

Figure 1 is a block diagram of an attribute map as used by an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Reference will now be made in detail to preferred embodiments of the invention, examples of which are illustrated in the accompanying drawings. Wherever convenient, the same reference numbers will be used throughout the drawings to refer to the same of like parts.

To introduce the AO (attribute-oriented) programming-language construct inventions, a generic class notation is first used to denote an example declaration from current-practice object-oriented programming: "There is a class Mammal with a method hair-length ()".

```
class Mammal is-subclass-of Animal
{
        hair-length ();
}
```

This class's one method is an observer referring to an attribute. The return type of the method is omitted for clarity.

Current-practice object-oriented programming allows only a highly restricted type of answer to the (evidently-redundant) programming question: "to what sorts of objects is the hair-length () method applicable?" The only type of answer allowed is, "to objects in the class Mammal" (i.e. objects that have attributes that put them in the Mammal class).

Attribute-oriented programming extends the possible types of answers to this question, allowing the applicability of methods to be more flexibly determined. To elaborate on this, additional notation is introduced. It is conceptually non-problematic to re-cast the class declaration above as:

```
class Mammal is-subclass-of Animal
{
}

Mammal hair-length ();
```

Which trivially divides the class declaration into, "There is a class Mammal", followed by, "Within the class Mammal, there is a method hair-length ()". Inspecting this alternative notation, it becomes evident that, along with the signature of a method, the computer programmer is also specifying a type of applicability-condition for the method. The applicability-condition can be described using the following notation:

```
(p : | p is-of-class Mammal).hair-length ();
```

This reads "for p such that p is of class Mammal, the method hair-length () is applicable". Thus the method-declaration has the form:

```
(<class-membership-predicate>).<method>(<args>);
```

Attribute-oriented programming allows other kinds of applicability conditions, so that in general a method-declaration can have the form:

```
(<applicability-predicate>).<method>(<args>);
```

The inventors term the parenthesized predicate an applies-if guard, and this technique of method-declaration is termed AO method-declaration.

Use of the AO method-declaration construct will be explicated in an example here by encoding the attribute map shown in Figure 1. (The following notation employs the "==" convention for denoting the equality predicate.)

"For p in general, is-animal () applies."
(p :| true).is-animal ();

"For p such that p.is-animal () holds, habitat () applies."
(p :| p.is-animal ()).habitat ();

And so forth...

(p :| p.is-animal ()).zooclass ();

(p :| p.zooclass () == mammal).hair-length ();

(p :| p.zooclass () == mammal).weaning-period ();

(p :| p.habitat () == water).in-saline-water ();

(p :| p.habitat () == water).mates-in-water ();

(p :| p.habitat () == water).diving-depth ();

Note that the applies-if guard "p.is-animal()" holds implicitly for the hair-length () method (e.g.), since the zooclass () attribute in the latter's applies-if guard is itself only applicable when "p. is-animal ()" holds.

The notation here is augmented to allow indication of the appropriate validity condition for each declared method, by the use of a valid-if clause. (Intuitively speaking, valid-if clauses specify the values that attributes can take on, or more generally the values that methods can return, and are a generalization of type specifications.)

"The value of is-animal () is a boolean value."

```
(p :| true).is-animal ():
        is-animal() == <boolean>;
```

"The value of habitat () is either land or water."

```
(p :| p.is-animal ()).habitat ():
        habitat () == land or habitat () == water;
```

And so forth...

```
(p :| p.is-animal ()).zooclass ():
        zooclass () == bird or zooclass () == mammal;

(p :| p.zooclass () == mammal).hair-length ():
        hair-length () == short-hair or hair-length () == long-hair;

(p :| p.zooclass () == mammal).weaning-period ():
        weaning-period () == <integer>;

(p :| p.habitat () == water).in-saline-water ()
        in-saline-water () == <boolean>;

(p :| p.habitat () == water).mates-in-water ()
        mates-in-water () == <boolean>;

(p :| p.habitat () == water).diving-depth ():
        diving-depth () == <integer>;
```

The use of the applies-if guard may be extended to another AO programming construct, the applies-if block. The applies-if block is used to collect methods that share a given applies-if guard:

```
(p :| true)
{
        is-animal () :
                is-animal () == <boolean>;
}

(p :| p.is-animal ())
{
        habitat ():
                habitat () == land or habitat () == water;

        zooclass ():
                zooclass () == bird or zooclass () == mammal;
}

(p :| p.zooclass () == mammal)
{
        hair-length ():
                hair-length () == short-hair or hair-length () == long-hair;

        weaning-period ():
                weaning-period () == <integer>;
}

(p :| p.habitat () == water)
{
        in-saline-water ():
                in-saline-water () == <boolean>;

        mates-in-watero :
                mates-in-water () == <boolean>;

        diving-depth ():
                diving-depth () == <integer>;
}
```

In some cases, all of the methods appearing within an applies-if block can be assumed to be taken with respect to the guard's object variable (e.g. "p" above). In these cases, an abbreviated notation may be used for the applies-if guard:

```
(true)
{
        is-animal ()
                is-animal () == <boolean>;
}

(is-animal ())
{
        habitat ():
                habitat () == land or habitat () == water;

        zooclass ()
                zooclass () == bird or zooclass () == mammal;
}

(zooclass () == mammal)
{
        hair-length ()
                hair-length () == short-hair or hair-length () == long-hair;

        weaning-period ():
                weaning-period () == <integer>;
}

(habitat () == water)
{
        in-saline-water ():
                in-saline-water () == <boolean>;

        mates-in-water ():
                mates-in-water () == <boolean>;

        diving-depth ():
                diving-depth () == <integer>;
}
```

It may be noted that there is some similarity between applies-if blocks and class declarations. This similarity exists because traditional class-hierarchy-usage is, in some sense, a special case of more generalized of AO programming. The AO-programming generalization of classes are applicability conditions, and the AO-programming generalization of class declarations are applies-if blocks. AO-programming is more general because, within the AO constructs, any configuration of attributes can provide the context within which methods are declared.

Other embodiments will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with a true scope of the invention being indicated by the following claims and equivalents.

WHAT IS CLAIMED IS:

1. A high-level computer language construct comprising:

an attribute; and

an applicability-predicate associated with the attribute, the applicability-predicate defining a range of entities to which the attribute applies.

2. A construct as recited in claim 1 wherein the attribute is a method declaration.

3. A construct as recited in claim 1 wherein the applicability-predicate has the form "for x such that x is of y" where x is a variable and y is a known range of entities.

4. A construct as recited in claim 3 wherein y is defined using an expression.

5. A construct as recited in claim 1 wherein the entities within the range of entities are class members.

6. A construct as recited in claim 5 wherein the attribute is a method declaration and wherein the construct defines the class members to which the method is declared.

7. A construct as recited in claim 1 that further comprises a valid-if clause, the valid-if clause defining a range of values that are associated with the attribute.

8. A construct as recited in claim 7 wherein:

the attribute is a method declaration;

the applicability-predicate defines a range of class members; and

the valid-if clause defines the values that the method may return.

9. A high-level computer language construct comprising:

a block including at least one attribute; and

an applicability-predicate associated with the block, the applicability-predicate defining a range of entities to which the block applies.

10. A construct as recited in claim 9 wherein the attributes in the block are method declarations.

11. A construct as recited in claim 9 wherein the applicability-predicate has the form "for x such that x is of y" where x is a variable and y is a known range of entities.

12. A construct as recited in claim 11 wherein y is defined using an expression.

13. A construct as recited in claim 9 wherein the entities within the range of entities are class members.

14. A construct as recited in claim 13 wherein the attributes in the block are method declarations and wherein the construct defines the class members to which the methods are declared.

15. A construct as recited in claim 9 wherein the block includes at least one valid-if clause, the valid-if clause defining a range of values that are associated with one of the attributes in the block.

16. A construct as recited in claim 15 wherein:

the attributes in the block are method declarations;

the applicability-predicate defines a range of class members; and

each valid-if clause defines the values that a respective method may return.

17. A method for writing computer programs using a high-level computer language, the method comprising the steps of:

defining an attribute; and

defining an applicability-predicate associated with the attribute, the applicability-predicate defining a range of entities to which the attribute applies.

18. A method as recited in claim 17 wherein the attribute is a method declaration.

19. A method as recited in claim 17 wherein the applicability-predicate has the form "for x such that x is of y" where x is a variable and y is a known range of entities.

20. A method as recited in claim 19 wherein y is defined using an expression.

21. A method as recited in claim 17 wherein the entities within the range of entities are class members.

22. A method as recited in claim 21 wherein the attribute is a method declaration and wherein the construct defines the class members to which the method is declared.

23. A method as recited in claim 17 that further comprises a valid-if clause, the valid-if clause defining a range of values that are associated with the attribute.

24. A method as recited in claim 23 wherein:

the attribute is a method declaration;

the applicability-predicate defines a range of class members; and

the valid-if clause defines the values that the method may return.

25. A method for writing computer programs using a high-level computer language, the method comprising the steps of:

defining a block including at least one attribute; and

defining an applicability-predicate associated with the block, the applicability-predicate defining a range of entities to which the block applies.

26. A method as recited in claim 25 wherein the attributes in the block are method declarations.

27. A method as recited in claim 25 wherein the applicability-predicate has the form "for x such that x is of y" where x is a variable and y is a known range of entities.

28. A method as recited in claim 27 wherein y is defined using an expression.

29. A method as recited in claim 25 wherein the entities within the range of entities are class members.

30. A method as recited in claim 29 wherein the attributes in the block are method declarations and wherein the method defines the class members to which the methods are declared.

31. A method as recited in claim 25, further comprising the step of defining at least one valid-if clause within the block, the valid-if clause defining a range of values that are associated with one of the attributes in the block.

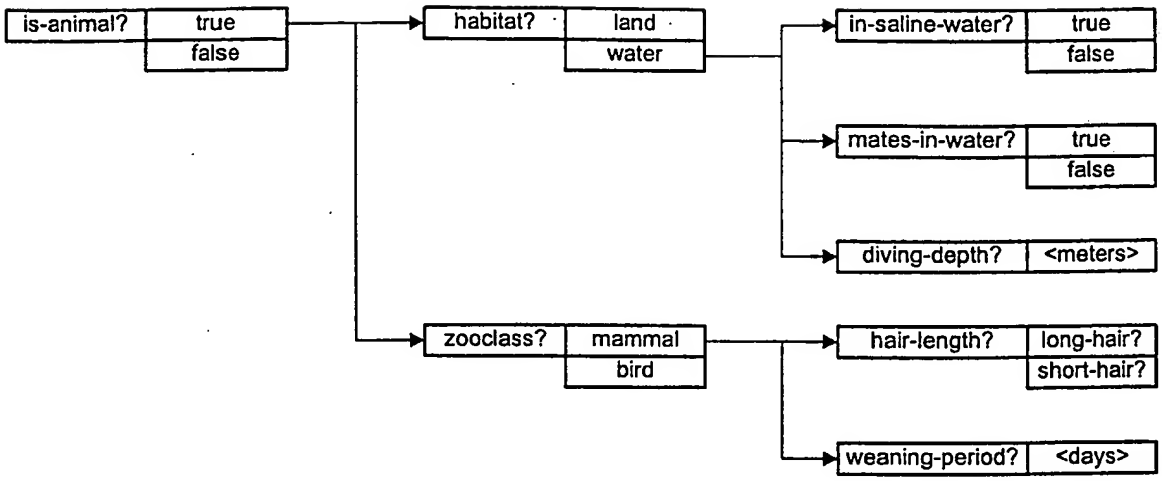
32. A method as recited in claim 31 wherein:

the attributes in the block are method declarations;

3 the applicability-predicate defines a range of class members; and

each valid-if clause defines the values that a respective method may return.

Fig. 1


is-animal?
true
false
habitat?
land
water
in-saline-water?
true
false
mates-in-water?
true
false
diving-depth?
<meters>
zooclass?
mammal
bird
hair-length?
long-hair?
short-hair?
weaning-period?
<days>

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/00490

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 9/445
US CL : 717/1
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 717/1, 8

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST, IEEE, ACM
Search terms: Toxonomic, predicate, attribute, grammar, software design, attribute-oriented

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	PAAKKI, Attribute Grammar Paradigms - A High-Level Methodology in Language Implementation, ACM Computing Serveys, June 1995, Vol. 27, No. 2, Pages 196-255, especially pages 200-202, 205-206, 212-237.	1-32
X	TAN et al., Coping with Changes in an Object Management System Based on Attribute Grammars, ACM, December 1990, pages 56-65, entire reference.	1-32
A	YANG et al., A Program Integration Algorithm that Accommodates Semantics-Preserving Transformations, ACM, December 1990, pages 133-143, entire reference.	1-32

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 MAY 2000	25 MAY 2000

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Tariq Hafiz
Facsimile No. (703) 305-3230	Telephone No. (703) 308-9049

Form PCT/ISA/210 (second sheet) (July 1998) *

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/00490

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	MCALLESTER et al., Taxonomic Syntax for First Order Inference, ACM, April 1993, pages 246-283, entire reference.	1-32
A	BRODSKY et al., Inference of Inequality Constraints in Logic Programs, ACM, May 1991, pages 227-240, entire reference.	1-32
A	KEMPER et al., Optimizing Disjunctive Queries with Expensive Predicates, ACM, May 1994, pages 336-347, entire reference.	1-32
A	CARTER et al., Efficient Attribute-Oriented Generalization for Knowledge Discovery from Large Databases, IEEE, April 1998, pages 193-208, entire reference.	1-32